THE ROLE OF MACHINE LEARNING IN BUILDING IMAGE INTERPRETATION SYSTEMS

TERRY CAELLI*

Department of Computer Science, Curtin University of Technology
Perth, WA 6001, Australia

WALTER F. BISCHOF†

Department of Psychology, University of Alberta
Edmonton, Alberta T6G 2E9, Canada

Machine learning has been applied to many problems related to scene interpretation. It has become clear from these studies that it is important to develop or choose learning procedures appropriate for the types of data models involved in a given problem formulation. In this paper, we focus on this issue of learning with respect to different data structures and consider, in particular, problems related to the learning of relational structures in visual data. Finally, we discuss problems related to rule evaluation in multi-object complex scenes and introduce some new techniques to solve them.

Keywords: Object recognition, machine learning, scene understanding, image labelling.

1. INTRODUCTION

In this paper, we discuss applications of machine learning (ML) techniques to the development and optimization of image interpretation systems. Image interpretation refers to the process of labeling image data with respect to a set of criteria that can either be known *explicitly* to the observer or system or may be encoded *implicitly* in the specific algorithms and constraints used by the system. In both cases, machine learning techniques are useful for making generalizations from training data, such as predicting labels for newly observed image data, for estimating parameters of sub-processes, such as parameters controling image segmentation, or for optimizing the procedures for searching and labeling images.

In the following, we discuss some general issues regarding the application of machine learning to image interpretation. We introduce different approaches to unsupervised and supervised learning and consider several fundamental issues regarding the representation of (a) image data, (b) knowledge required to interpret image data, and (c) rules used for interpreting image data. Next, we study the application of machine learning in low-level and high-level vision and discuss evaluation criteria and design issues that arise at both levels. Finally, we present several systems that were designed to learn image interpretation rules, both in low-level and high-level vision applications.

*Supported by a grant from the Australian Research Committee.
†Supported by Grant OGP38521 from the Natural Sciences and Engineering Research Council of Canada.

International Journal of Pattern Recognition and Artificial Intelligence Vol. 11 No. 1 (1997) 143–168

1.1. Learning Paradigms

A representative sample of general ML techniques already in use for image interpretation is shown in Table 1. The techniques can be grouped into "supervised" and "unsupervised" learning. In the case of supervised learning, the output states of the image interpretation system (for example, the labels to be assigned to different image parts) are known explicitly and the system is aimed at predicting these states. In the case of unsupervised learning, the output states are defined only implicitly in the specific algorithms and constraints that are used.

Table 1. Representative machine learning techniques for computer vision.

	Unsupervised learning Model	Major application
Parametric	Bayesian classifiers	Segmentation
	Radial basis functions	Image synthesis
Non-parametric	K-means	Feature grouping
	Kohonen maps	Segmentation
	Vector quantization	Image compression
	COBWEB	Visual taxonomies
	Supervised learning Model	**Major application**
Attribute-indexed	Discriminant functions	Classification
	Decision trees	Recognition
	Neural networks	Feature extraction
	Evidence-based systems	Recognition
Part-indexed	Rulegraphs	Recognition by parts
	Conditional rule generation	Recognition by parts
	FOIL	Symbolic descriptions

Unsupervised Learning. Classic examples of unsupervised learning models include clustering techniques where input data is grouped into sets of similar examples using proximity analysis of example attributes. The result of the grouping process depends on the particular characteristics of the proximity models. Both parametric and non-parametric techniques have been used extensively. Parametric techniques include parametric Bayesian clustering where (probabilistic) cluster membership is modeled using, for example, multivariate Gaussian functions.[1] Alternatively, the functions can be used to synthesize observed data (images), as is the case with the radial basis function formulation.[2] In both cases, clustering involves determining position and shape parameters of the best fitting functions for modeling the clusters, and the algorithms for finding them typically use gradient descent with respect to the function parameters.

Parametric clustering techniques have several drawbacks, including the complexity of the search procedure required, the lack of unique solutions and, most importantly, the assumption that such functions actually do represent the data. For these reasons, non-parametric clustering techniques have been more popular. Clustering is aimed at partitioning data such that within-cluster distances are minimized and between-cluster distances are maximized. The well-known methods shown in Table 1 share common features to achieve this goal. For the K-means method, the formulation is direct in that the search for clusters is guided by the minimax constraint (i.e. minimizing within-cluster distances and maximizing between-cluster distances). Kohonen maps enact a similar process[3] through the formation of attractors in the data-attribute space. In vector quantization[4] clustering is achieved through binning techniques[5] while conceptual clustering systems like COBWEB[6] create partitions in attribute space by maximizing category utility measures.

Supervised Learning. Supervised learning differs from unsupervised in so far as criteria for labeling data (e.g. classification labels) are known explicitly. Given some training data described in terms of a set of features (attributes) and their class labels, the goal of supervised learning is to find a (simple) partitioning of the feature (attribute) space that allows correct classification of all training data, as well as generalization from training data to unseen, similar data. Supervised learning methods differ with respect to the feature space partitioning they produce, as illustrated in Fig. 1. Given that each region in the partition defines an area of generalization, the methods therefore differ with respect to the produced generalizations.

One class of procedures partition the attribute space using perceptron-like linear decision functions. These include classical discriminant function methods,[7] linear decision trees and decision trees based on neural networks.[8] Elaborations of these methods involve neural network-based approaches such as cascade-correlation,[9] radial basis functions,[2] or vector quantization.[4]

A second class of methods, rule-based methods, also partition attribute spaces with respect to class labels but the partition boundaries are constrained to orientations parallel to the axes. This is done to allow extraction of simple rules of the form

```
if <conditions on feature states>
then <evidence weights for each class>
```

where the rule condition is defined by a conjunction of attribute bounds (and associated labels) and the rule actions are defined in terms of fuzzy class memberships, Bayesian posterior probabilities, or weights in a neural network.[10] The more popular rule-based methods include decision trees[11] and evidence-based systems.[12,10]

1.2. Representation

Image information can be represented in at least two different schemes. In one scheme, patterns are encoded as images, per se, in an image coordinate system (Cartesian, log-polar, or coordinates of transforms such as Fourier or Hough) with

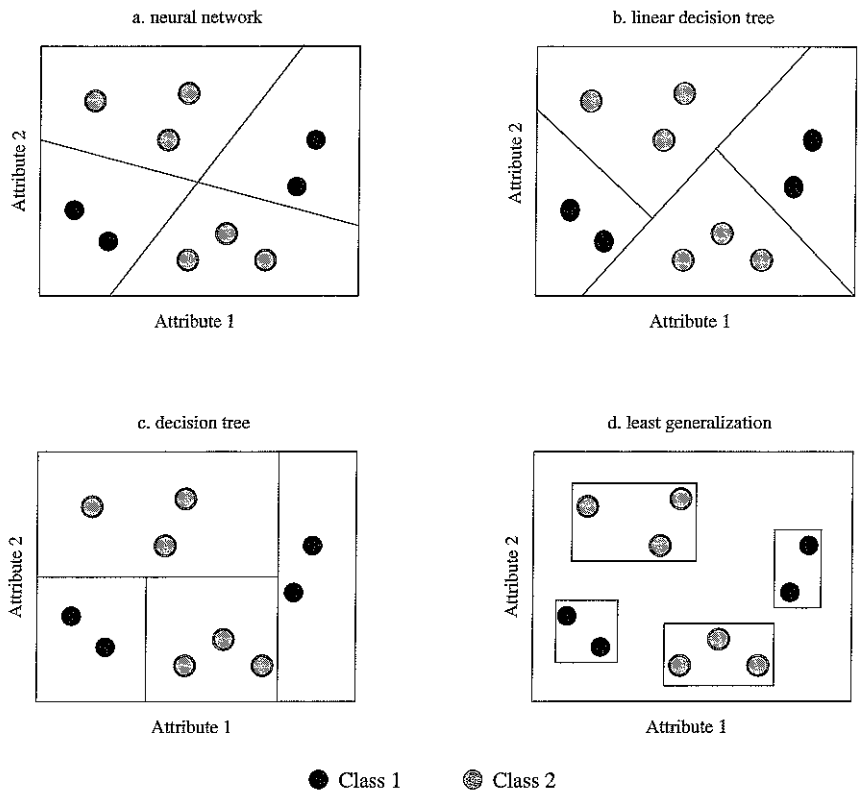


Fig. 1. Feature space partitioning produced by four different methods. (Top left) Neural network: Lines correspond to zeros of perceptron-type (hidden) units. (Top right) Linear decision tree: For each polygonal region splits are introduced recursively to maximize class evidence. (Bottom left) Decision tree: Splits are made at orientations orthogonal to feature axes. They maximize class evidence in each region and introduce an ordering of attributes. (Bottom right) Least generalization: Bounding boxes determine limits for class evidence.

attribute values (e.g. intensity, color, amplitude, phase, or frequency) defined at each coordinate. In the other system, patterns are defined in terms of coordinate values in an abstract attribute (sometimes called 'feature') space. The feature space is not necessarily related directly to the initial encoding of the image data and may include attributes of pixels, of pattern parts, or of complete patterns.

In this latter representation an image composed of multiple parts can be described in terms of (unary) features of single parts, as well as (binary) features of relations between different parts. For example, the parts of an image may correspond to segmented image regions, unary features may include area or average brightness of each region, and binary features may include center–center distance or length of shared boundaries between different regions. In some of the representations, it is made explicit that a given unary or binary relation holds for specific parts (e.g. "Image part 12 and image part 15 have a center–center distance of 300 pixels"), i.e. the connection between unary and binary attributes and labels of pattern parts

is made explicit. In other representations, this information is not represented (e.g. "Two of the image parts have a center–center distance of 300 pixels"). The former representation is referred to as a *part-indexed* or label-compatible representation, whereas the latter is referred to as an *attribute-indexed* representation. In general, part-indexed representations have a greater representational power but are inferior to attribute-indexed representations with respect to the efficiency of the associated matching procedures.[13]

Attribute-indexed systems rely on a single representational domain, the attribute domain. Part-indexed systems can rely also on information about image coordinates due to the fact that part label information is preserved. This allows part-indexed systems to operate in a joint image-attribute domain. In a sense, this is similar to classical Gabor-representations of spatial information[14] in that they generate representations based on minimal descriptions in terms of image parts while, at the same time, minimizing joint occurrences of different classes in attribute space (corresponding to Gabor representations minimizing joint uncertainty in the image and attribute domains).

2. MACHINE LEARNING IN LOW-LEVEL VISION

In discussing applications of ML to low-level vision, we restrict our attention to edge extraction and region segmentation, even though ML has also been used in other areas, such as image restoration or image enhancement.[15] Approaches to low-level vision involve the development of operators for the detection of features such as edges, corners or lines using filter-based, statistical or geometric techniques. Most *classical approaches* have been essentially "open-loop" in the sense that the output of an operator is not used to estimate or adjust operator parameters, or, in general, that image-dependent statistics are used to optimize these parameters. For example, although edge operators at multiple scales produce evidence for different types of image gradients or edges the resultant edge maps are typically not used to adjust the range of scales used.[16] In other words, such an approach does not "solve" a clear problem such as determining edges that are acceptable to a human observer or edges that satisfy particular physical constraints.

In *adaptive signal analysis*, on the other hand, operator parameters are estimated and adjusted to optimize image-dependent constraints. For example, parameters of edge detectors can be adjusted so as to maximize, in a statistical sense, the explained image variance in local image regions.[17] However, the design characteristics of the operators remain fixed and specific to given images. It is desirable that image operators can be trained to produce outputs that correspond to known features as close as possible and that the resultant operators can be applied to new data, i.e. that they can generalize from the training examples. This is precisely what can be achieved using ML approaches.

In recent years there has been considerable interest in using *connectionist learning procedures* for low-level vision. Specifically, feed-forward neural networks

have been used to learn edge extraction, image enhancement and texture segmentation.[15,18] They can be used in both supervised and unsupervised modes to determine the types of pixel window values which evidence different feature types. Indeed, it can be shown that hidden layers of a neural network can compute both stationary or non-stationary adaptive filters depending on how the weight matrices are formulated in the formation of the filter point-spread functions.[19] The development of non-stationary adaptive operators, i.e. of operators whose parameters can vary for different image regions, constitutes a major development over past models which were based on classical stationary signal processing even though they may have been adaptive.

Recently, we have extended this type of machine learning approach to low-level vision using a *"model-based" neural network* (MBNN) approach.[20] MBNNs differ from usual neural networks in that connection weights as well as the (non-linear) transducer functions are parameterized and modeled in such a way to guarantee an output which satisfies given constraints. For example, we have developed modules of MBNNs which can compute eigenvectors or quantize histogram data. This type of network is illustrated in Fig. 2 which has been developed for encoding the degrees to which patterns are invariant to rotations, shift and scale, i.e. for encoding the invariance signature of a shape. Such signatures have recently been used successfully in invariant optical character recognition.[20]

We argue that machine learning techniques have significant uses in the development of robust low-level segmentation and feature extraction processes. However, they must take into account the following constraints. First, input to the ML models must reflect desired output properties, such as, for example, rotation invariance of operators. Second, the ML models must be capable of operating directly on image pixels and regions, preferably in a parallel fashion and at multiple scales. Third, cost functions and model parameters should be chosen to reflect domain knowledge. Finally, we believe that model-based and modular ML systems are particularly promising given that their behavior can be analyzed more clearly and learned rules can be more easily described than is the case with standard neural nets.

3. MACHINE LEARNING IN HIGH-LEVEL VISION

High-level vision is concerned with the interpretation of patterns and images, in the sense of determining classifications and other labels for image and pattern parts, and possibly determining mappings between given patterns and stored models to determine the pose of objects or to predict position and existence of occluded pattern parts. Machine learning has received most attention in this area, given that — *ab initio* — classical pattern recognition problems have been posed in terms of supervised learning and classification. More recently, analytic parametric classifiers, such as linear discriminant functions, have been replaced by methods based on multiple evidential systems and search techniques such as neural networks, least squares and nearest neighbor algorithms. The problem remains, however, the same.

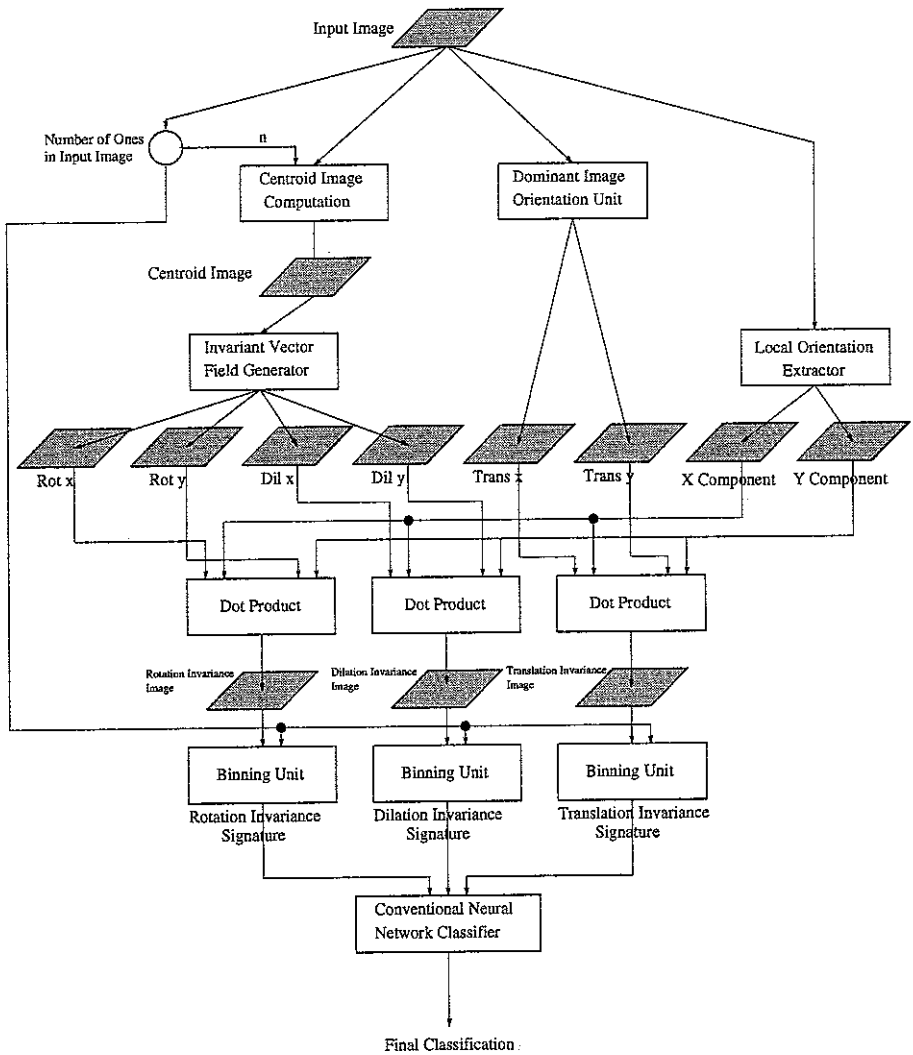


Fig. 2. The idea of model-based neural networks is to constrain or model the weights in such a way that, at particular levels of processing, different operations are guaranteed to occur. In this example, the extraction of directional derivatives of curves, their invariance signatures are used to recognize patterns invariant to rotations, translations and scale. Here, the network computes the products of local curve orientations ($X, Y components$) with the vectors corresponding to invariant vector fields for rotations (Rot x, y), dilations (Dil x, y) and translations (Trans x, y). The histograms of such scalar products are computed, quantized and then used as input to a classifier — all within a standard feed-forward neural network architecture (see Ref. 20).

That is, we are concerned with the problem of identifying known and novel patterns or structures in simple or complex ("scene") image data and the power of such systems lies in their ability to generalize from training data while still retaining classification accuracy.

3.1. Design Issues

The design and application of ML techniques in high-level vision revolves around the four major issues, pattern decomposition, feature selection, rule generation and rule application.

Pattern Decomposition. Patterns have been typically represented by one of three methods. The first, *template representation* encodes the pattern by an image, per se, albeit normalized, pre-processed for edges and related features and at multiple scales. The major problems with such representations are that they are not rotation and scale invariant nor lack the ability to readily encode the required variability in structure. The second, *global feature representation* uses features of image transforms to encode shape.These include Fourier, Hough and moment generating function-based transformations. Typical problems with these methods include their lack of invariance, uniqueness with limited numbers of components and their derivation from the complete image: they do not apply to complex multiple-object scenes. The third, *encoding-by-parts* represents patterns by their labeled parts, relations and their associated unary and binary attributes. Though dependent on segmentation/feature extraction processes, we argue that this method proves to be the most useful for invariant pattern recognition, the generation of structural descriptions of shape and the recognition of patterns in complex scenes.

Rule Generation and Application. Whether the decomposition is defined by templates, global image transforms or part-decompositions, the issues related to feature/attribute selection and generalization have common characteristics and involve some type of learning strategy. As discussed in Sec. 1, supervised learning methods differ with respect to the features space partitioning model employed (Fig. 1). Linear discriminant functions, linear decision trees, Voronoi tesselations produce different types of partitions which, in turn, determine different types of pattern "rules" or "generalizations" from training data. On the other hand, techniques which generate rules with Horn clause conditions (conjunctions of attribute bounds), so-called "rule-based methods", impose constraints on the partition boundaries. This is done to enable rapid rule evaluation and to allow the derivation of symbolic descriptions, even if it is achieved at the cost of obtaining less efficient feature space partitions.

Application of classification rules that are acquired in a supervised learning paradigm is complicated by several factors. In typical applications, the number of rules and/or classes is large and exhaustive search through all classification rules may not be feasible. Schemes for *efficient rule evaluation* have to be devised that may include pre-compiling of search strategies or the use parallel search methods.

Recognition from partial data: Many real recognition problems involve the recognition of patterns from partial data. We claim that in order to optimize this process, rule generation and recognition methods are required that are based on a "recognition-by-parts" approach. In this approach, pattern fragments or individual pattern parts (and their associated attributes) are learned to provide sufficient evidence for the classification of patterns.

Recognition from distorted data: Most realistic recognition problems involve identification of known classes or patterns in data which is not identical to any of the training data. Solutions involve the ability of learning systems to generalize from training data. The problem is to optimize generalization while retaining recognition accuracy.

Finding patterns in complex data: Perhaps one of the more difficult problems in recognition is that of finding patterns (or objects) in complex multiple object scenes. Solutions must involve evidencing distinct pattern regions in an efficient way. One may find efficient solutions in specific applications but, in general, the problem is ill-posed.

Correspondence between data parts and model parts: Beyond a mere classification of patterns or objects, it may be required to determine the correspondence between the parts of a pattern and those of stored model patterns. This may be required to determine the pose of objects or in order to predict the existence of (hidden) pattern parts. Solutions to this problem depend on whether learning or encoding training data also includes pattern or object part labels. Part-indexed data structures implicitly solve the pose problem while simple attribute-indexed systems, such as evidence-based systems[12] do not. These methods may have to be supplemented by methods for additional hypothesis testing or model projection to solve the pose problem. This will be explored further in the following sections.

3.2. Non-Inductive Recognition Systems

Most recent recognition systems are based on the recognition-by-parts approach. This allows the (range- or intensity-based) encoding of objects by parts over different views. Further, global features are rarely appropriate for highly variable or for fragmented 2D patterns: Fourier or Karhunen–Loéve transforms of complete patterns cannot easily be used to identify fragments nor find patterns embedded in complex scenes. For these reasons, most recent recognition systems assume the recognition-by-parts representation where pattern and data are in the form of labeled graphs or labeled and attributed graphs. Consequently, methods are required for efficient instantiation of known model graphs in observed data.

One group of methods, including those developed by Grimson[21] and Bunke,[22,23] use, essentially, traditional graph matching methods for solving correspondences between model and data graphs, and rely on constraint propagation to improve the efficiency of the matching procedure. An alternative approach is geometric hashing: unary and binary features of model parts and their relations are stored in a hash table where each entry consists of an n-tuple of feature states and a list of training patterns that satisfy these feature states.[24]

From a machine learning perspective, there are several problems with these non-inductive approaches. They do not explicitly deal with generalization (induction) in the sense that they do not determine parts, relations and their associated attribute bounds that are necessary and sufficient for recognizing training patterns and similar, unseen patterns. They do not typically use part indexing as a constraint in

compiling the hash functions. However, they "pre-compile" the ways in which parts or attributes need to be checked for efficient evaluation of different models.

3.3. Inductive Recognition Systems

Induction or generalization is concerned with summarizing data in such a way that the resultant indexing system can efficiently evidence different classes or patterns. ML is involved in generalization, in the selection of parts and attributes, and the optimization of the resulting rules.

Many methods reduce encoding and recognition processes to creating rules or tables where specific feature values are used to index classes of patterns that were present in training data. Machine learning is concerned with issues related to the form of such rules, their automatic generation, and their optimization. There are many different systems to attain these goals, and one of the aims of this paper is to point out domains where these systems work and where they fail.

An important issue is the amount of relational information that is needed to encode patterns. This is important for the development of robust and efficient recognition systems in both 2- and 3-dimensional domains. As discussed before, representational schemes can be classified as either attribute-indexed or as part-indexed. In part-indexed representations, connections between unary and binary features or predicates are made explicit, and structural pattern information is thus completely preserved. In attribute-indexed representations this connection is lost and, consequently, structural pattern information can be represented only to a limited extent. In the following sections, we present first an example of an attribute-indexed inductive system, followed by two different types of part-indexed inductive systems.

3.4. Attribute-Indexed Inductive Systems: EBS

In many applications, data is essentially non-relational and attribute-indexed systems are perfectly adequate. This is the case with many of the datasets used in the machine-learning literature. It is also true in spatial computing, remote sensing, or in image classification problems, where classifications are often based on simple feature vectors, such as intensity, color or spectral values. It also applies to pattern and object recognition models where morphological or global attributes are used to define shapes. Furthermore, attribute-indexed systems are capable of encoding a limited amount of structural information, as discussed below.

Evidence-based systems ("EBS") are an example of attribute-indexed systems that have been used for 3D object recognition. We have developed an EBS for range-based object recognition[10] as an extension of earlier work by Jain and Hoffman.[12] In this approach, learning of a set of training images or patterns is achieved in three steps: feature extraction, rule generation, and evidence combination.

Feature Extraction. In a first step, each image or pattern of a training set is segmented into parts. Pattern features are extracted that include global features of

the whole pattern, unary features of pattern parts, and binary features of relations between pattern parts. The features are collected into three separate feature spaces. A point in the unary feature space, for example, represents a feature vector of a single pattern part, and is labeled with the class of the corresponding pattern.

Rule Generation. Given that all training patterns are encoded in multiple feature spaces, an attempt is made to capture the predominant characteristics of the training samples by grouping them into spatially-delimited regions or clusters in feature space. The bounds on such regions can be used as conditions for the activation of rules, and they define the degree of generalization from samples. Each rule has associated evidence-weights that correspond to the likelihood that activation of the rule contributes evidence for the existence of an object or class of objects.

Generation of rules and evidence weights involves the use of clustering algorithms. For reasons discussed above, such clusters are defined by hyper-rectangles and oriented along the feature space axes to allow for rules of the conjunctive form. Since clusters need not be disjoint, more complex definitions of rule conditions can be constructed. For example, non-convex regions can be defined logically by rules which include some regions of feature space but explicitly exclude others.

In the EBS system developed by Jain and Hoffman,[12] rules were generated by clustering the samples in feature space using a minimum spanning tree technique. In our work,[25] we have used minimum entropy clustering which adjusts the position and size of a fixed number of rectangles (clusters) to maximally separate the occurrences of class samples per cluster. In other words, we relabel the cluster membership of each sample to minimize the entropy function:

$$H_{min} = \min_{i \in j} \left\{ -\sum_{j} \sum_{k} p_{jk} \ln p_{jk} \right\} \tag{1}$$

where p_{ij} is the probability of class i occurring in cluster j and the probability is determined from the relative frequency of class samples within a given cluster solution.

Evidence Combination. All the evidence provided by the global, the unary, and the binary features must be combined into a composite evidence value. Rather than use a predetermined scheme for evidence combination, we use supervised learning to estimate optimal weights of features, and, at the same time, to learn the relationships between unary and binary features. To achieve this, we have used a neural network with input nodes corresponding to activated rules (defined by the presence of given part and part relational attributes) which are combined during training via connection weights to maximize known classifications of objects (see Fig. 3). Generalization comes from two sources, first, the cluster bounds in attribute spaces determined by the clustering procedure, and, second, the degrees to which different observed attributes of parts and their relations can activate network weights

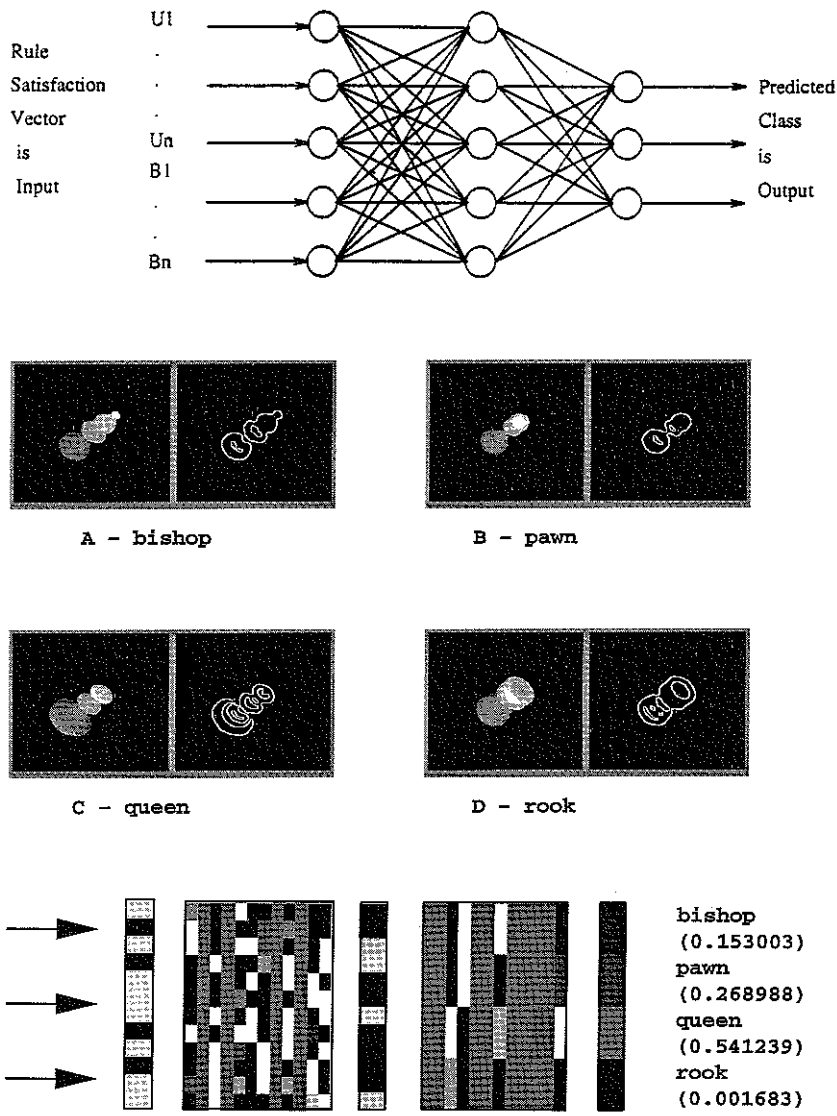


Fig. 3. Object recognition using an evidence-based neural network system. (Top) Rules are generated for objects involving the presence of part (unary feature bounds) and relational attributes (binary feature bounds) whose evidence weights are determined via a neural network. (Bottom) New views are classified according to the rules activated by the presence of parts and relations and evidence weights propagated via the network hidden units (from Ref. 10).

to result in the same class evidence vector. Figure 3 shows the case for classifying different chess pieces using this method. For both training and testing, input range data was initially partitioned in terms of convexity, concavity and planarity. For these parts, attributes were computed that included unary attributes such as curvature or area and binary attributes such as angle between surface normals or distance. These attributes were clustered using a minimum entropy method and

formed the (0,1) input to a simple feed-forward neural network to estimate the weights of such evidence in recognizing objects. The actual weights and states of the hidden units during the recognition phase are shown at the bottom of Fig. 3.

The evidence weights are determined by the connections between input, hidden, and output layer nodes. Each hidden layer node is connected to every unary and binary cluster. This allows for the reinforcement of co-occurrences between unary and binary feature states and thus allows implicit learning of relational structures. It does, however, not guarantee unique representation of structural relations.

EBS is an efficient and powerful method for learning classifications of complex patterns. Its main limitation is the fact that structural representation is not unique, i.e. rules are generated without *explicitly* considering the relationships between specific unary and binary feature values that define specific objects. Limited representation of structural information is attained *implicitly* via the hidden units in the neural network, but it does not *guarantee* a unique representation of structural relations in the data.

There are two principal ways to improve EBS-like systems with respect to the ability to represent structural pattern information. The evidence mechanism can be supplemented by a posterior test of structural pattern identity. Alternatively, evidence rules can be elaborated to contain (*a priori*) structural pattern information. The first path is chosen in the rule graph method,[26] and the second is chosen in the conditional rule generation (CRG) method.[13]

3.5. Part-Indexed Inductive Systems: Rulegraphs

The idea behind rulegraphs[26] is to use EBS evidence weights together with *explicit* structural pattern information to prune the search space in matching model graphs to data graphs. The technique relies on two simple principles: First, sets of model graphs and their vertices are reduced by generalization, collecting like features for different classes into clusters in feature space. Second, search for subsets of compatible labels between rules is constrained using evidence weights produced by an EBS. The matching process involves graphs of cardinality no greater than the number of unary rules and thus is more efficient than classical graph matching procedures.

A rulegraph is a graph of rules in which vertices correspond to unary rules and edges correspond to binary rules according to the following connection criterion: two unary rules U_i and U_j are connected by a binary rule B_k if there exist labels X, Y such that $X \in U_i$ and $Y \in U_j$ and $XY \in B_k$ (see Fig. 4).

The basic idea of graph matching using rulegraphs is illustrated in Fig. 4. Given a set of training patterns, a set of unary and binary rules are generated (Fig. 4(a)). Unary and binary rules that share common labels are connected by a rulegraph edge, according to the connection criterion (Fig. 4(b)). When a new pattern is presented to the system, several rules are activated (Fig. 4(c)), each evidencing different parts of the training patterns: Sample part X could correspond to parts A or D, Y could correspond to B, C, E, or F, and the binary relation $B(XY)$ could correspond to $B(AC)$ or $B(EF)$. Among these, only one interpretation is consistent with the

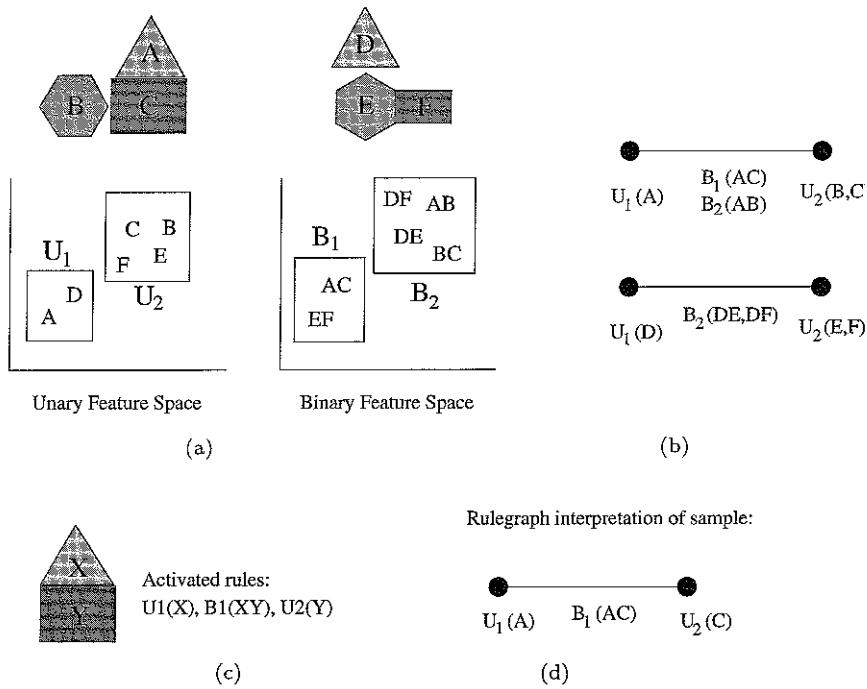


Fig. 4. Training patterns are used in (a) to label the unary and binary rules according to the mapping of the parts and their relationships into each feature spaces. Unary rules are labeled with single labels and binary rules are labeled with label pairs. Rulegraph models may then be formed, according to the connection criterion, and these are shown in (b). At run time, parts in the sample pattern activate unary and binary rules based on their feature states as shown shown in (c). The search for label-compatible rules between the sample and the model results in rulegraph interpretation (best match) as is seen in (d) (from Ref. 26).

rulegraph model, namely the label mapping $X \rightarrow A$ and $Y \rightarrow B$, and this is the interpretation accepted by the rulegraph system.

In rulegraphs, several labels may exist in each rule vertex and this gives rise to multiple mapping states involving the same labels. To determine label-compatibility between *rules* instead of *parts* we use a method which proceeds in five steps. In Step 1, all possible mapping states are created for all the labels in the sample and model rules. In Step 2, all incompatible label mappings between sample rulegraph and model rulegraphs are eliminated. In Step 3, the (multiple) remaining mapping states are updated by instantiation (if the label is *not* yet mapped) or elimination (if the label *is* mapped) using the mappings generated in Step 2 and the old mapping states. In Step 4, the mapping states are updated in order of decreasing evidence of rules into which they map. This ensures that labels which have strongest evidence for a particular class will be mapped first. Finally, in Step 5, we check that at least one binary rule is satisfied.

This method offers a technique for checking compatibility between rules. The problem of finding the *best* match now reduces to that of finding the largest evidenced set of rules which are all pairwise compatible. The cardinality of the search

problem has already been reduced to the number of unary rules instead of the number of primitive parts. Furthermore, the evidence weights can be used to direct the search toward rules and models for which strong evidence exists. To achieve this, we use A* search combined with the Bayesian evidence weight metric to allow pruning of the search tree (see Ref. 26).

3.6. Part-Indexed Inductive Systems: CRG

The idea of Conditional Rule Generation (CRG) is to generate classification rules that include structural pattern information to the extent that is required for classifying correctly a set of training patterns. CRG searches for the occurrence of unary and binary features states between connected pattern components and creates a tree of hierarchically organized rules for classifying new patterns. Generation of a rule tree proceeds in the following manner.

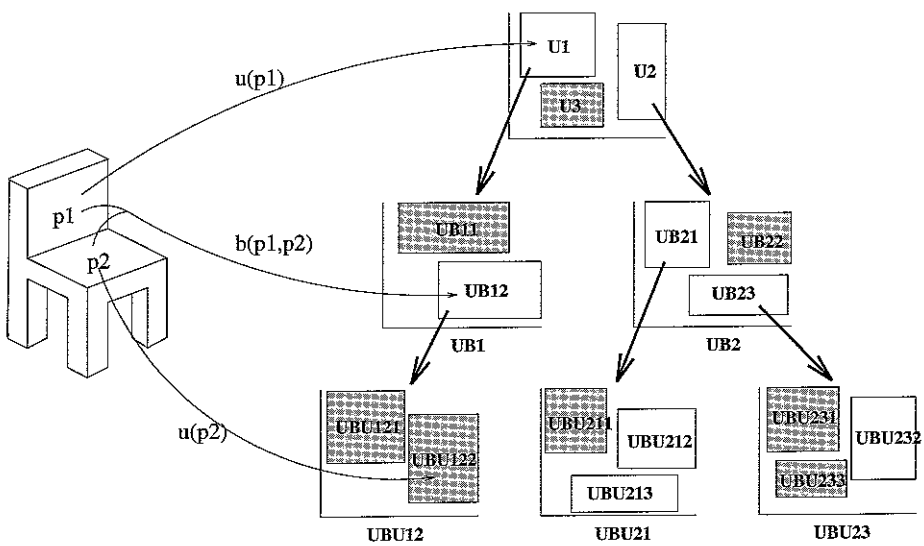


Fig. 5. Cluster Tree generated by the Conditional Rule Generation Procedure (CRG). The unresolved unary clusters ($U1$ and $U2$) — with elements from more than one class — are expanded to the binary feature spaces $UB1$ and $UB2$, from where clustering and expansion continues until either all rules are resolved or the predetermined maximum rule length is reached, in which case rule splitting occurs. Associated graphs are illustrated.

Cluster Tree Generation. First, the unary features of all parts of all patterns are collected into a unary feature space U in which each point represents a single part. The feature space U is partitioned into a number of clusters U_i. Some of these clusters may be unique with respect to class membership (e.g. U_3 in Fig. 5) and provide a classification rule: If a pattern contains a part p_r whose unary features satisfy the bounds of a unique cluster U_i then the pattern can be assigned a unique classification. The non-unique clusters contain parts from multiple pattern classes and have to be analyzed further. For every part of a non-unique cluster (e.g. U_2 in

Fig. 5) we collect the binary features of this part with all other parts in the pattern to form a (conditional) binary feature space UB_i. The binary feature space is clustered into a number of clusters UB_{ij}. Again, some clusters may be unique (e.g. cluster UB_{11} in Fig. 5) and provide a classification rule: If a pattern contains a part p_r whose unary features satisfy the bounds of cluster U_i, and there is an other part p_s, such that the binary features of the pair $\langle p_r, p_s \rangle$ satisfy the bounds of a unique cluster UB_{ij} then the pattern can be assigned a unique classification. For non-unique clusters, the unary features of the second part p_s are used to construct another unary feature space UBU_{ij} that is again clustered to produce clusters UBU_{ijk}. This expansion of the cluster tree continues at additional levels $UBUB$, $UBUBU$, ... involving additional pattern parts until all clusters are completely resolved. Some clusters may, however, never be resolved. In this case, the cluster tree has to be refined by either re-clustering one of the features spaces or by splitting one of the clusters.

Feature space clustering can be obtained using parametric or non-parametric clustering, as discussed in the Introduction, or using fuzzy clustering.[27] Alternatively, one can rely completely on the cluster refinement methods discussed in the next section.

Cluster Tree Refinement. Analysis of non-unique clusters can proceed by further expanding the cluster tree, analyzing unary and binary attributes of additional pattern parts. Alternatively, the derived clusters in the tree can be refined or broken into smaller, more discriminating feature bounds or rules as described below. Both approaches have their respective disadvantages. Cluster refinement leads to an increasingly complex feature-space partitioning and thus may reduce the generality of classification rules. Cluster-tree expansion, on the other hand, successively reduces the possibility of classifying patterns from partial data. In the end, a compromise has to be established between both approaches.

One successful approach to cluster tree refinement involves entropy-based splitting procedures. Consider splitting the elements of an unresolved cluster C along a (unary or binary) feature dimension F. The elements of C are first sorted by their feature value $f(c)$, and then all possible cut points T midway between successive feature values in the sorted sequence are evaluated. For each cut point T, the elements of C are partitioned into two sets, $P_1 = \{c \mid f(c) \leq T\}$ with n_1 elements and $P_2 = \{c \mid f(c) > T\}$ with n_2 elements. We define the partition entropy $H_P(T)$ as

$$H_P(T) = (n_1 H(P_1) + n_2 H(P_2))/(n_1 + n_2).$$

The cut point T_F that minimizes $H_P(T_F)$ is considered the best point for splitting cluster C along feature dimension F. The best split of cluster C is considered the one along the feature dimension F that minimizes T_F.[5] For an unresolved leaf cluster C_L, one can split C_L or any cluster in the parent chain of C_L. Among these clusters, the one that minimizes T_F is considered the optimal point for refining the cluster tree.

A completely resolved cluster tree provides a set of deterministic rules for classification of patterns. In addition, partial rule instantiations (such as, for example, the classification associated with cluster UB_{23} in Fig. 5) can provide partial evidence for class membership of incomplete or distorted patterns.

Evidence Combination. Every cluster element in the cluster tree corresponds to a sequence $U_i - B_{ij} - U_j - B_{jk} \ldots$ of unary and binary features associated with a non-cyclic chain of pattern parts. CRG thus produces classification rules for (small) pattern fragments and their associated unary and binary features whereas EBS and rulegraphs produce classification rules for sets of unary and binary features. When CRG rules are applied to some pattern, one obtains one or more (classification) evidence vectors for each pattern fragment. These evidence vectors have to be combined into a single evidence vector for the whole pattern. Difficulties arise when a test scene contains multiple patterns where it is unclear whether a sequence $p_i - p_j - \ldots - p_n$ of pattern parts belongs to the same pattern or whether it is "crossing the boundary" of different patterns. In the former case, CRG rule can be expected to produce correct classifications, whereas in the latter case classification may be arbitrary.

In this section, we present a simple scheme for evidence combination in CRG-type schemes for classification of pattern fragments. In the next section, we will analyze more complex schemes for classifying scenes containing multiple patterns or objects.

Pattern identification and classification begins with the extraction of all non-cyclic paths up to a certain length l. These paths, termed *chains*, constitute the basic units for pattern classification. A chain is denoted by $S = \langle p_i, p_j, \ldots, p_n \rangle$ where each p_i denotes a pattern part. For some chains, all parts belong to a single pattern, but other chains cross the boundary between different patterns.

Each chain $S = \langle p_i, p_j, \ldots, p_n \rangle$ is classified using the CRG classification rules. Depending on the unary and binary feature states, a chain may or may not instantiate one (or more) classification rules. In the former case, rule instantiation may be partial (with a non-unique evidence vector $\vec{E}(S)$), or complete. As discussed above, the evidence vector for each rule instantiation is derived from the empirical class frequencies of the training examples.

The evidence vectors of all chains $\langle p_{i_1}, p_{j_1}, \ldots, p_n \rangle$, $\langle p_{i_2}, p_{j_2}, \ldots, p_n \rangle, \ldots$ terminating in p_n must be combined to obtain a classification for part p_n. This is a problem related to "stacked generalization",[28] with the added difficulty that some of the evidence vectors may be mutually incompatible.

A simple scheme for evidence combination is the *winner-take-all* solution: the evidence vectors of all chains terminating in p_n are averaged to give $\vec{E}_{av}(p_n)$, and the most likely class label is used to classify part p_n. This solution does not take into account that, for a chain $S = \langle p_i, p_j, \ldots, p_n \rangle$, the average evidence vectors $\vec{E}_{av}(p_i)$, $\vec{E}_{av}(p_j), \ldots, \vec{E}_{av}(p_n)$ may be very different and possibly incompatible. If they are very different, it is plausible to assume that the chain S is crossing a boundary. In this case, S and its evidence vectors should be disregarded.

Compatibilities between evidence vectors are taken into account in the *relaxation labeling* solution. Here, the weight of an evidence vector $\vec{E}(p_i)$ of part p_i depends on the similarity (compatibility) of $\vec{E}(p_i)$ to the evidence vectors of neighboring parts. The constraints on evidence vectors are propagated throughout the pattern using a standard relaxation labeling technique.[29] More precisely, the relaxation-labeling solution is given by

$$\vec{E}^{t+1}(p_i) = \Phi\left[\sum_{S=(p_i \ldots p_n)} \vec{E}^t(p_i) C(p_i, p_n)\right] \tag{2}$$

where $\vec{E}^t(p_i)$ corresponds to the evidence vector of p_i at iteration t, with $\vec{E}^0(p_i) = \vec{E}_{av}(p_i)$. Further, $C(p_i, p_n)$ corresponds to the compatibility between parts p_i and p_n, and Φ is the logistic function

$$\Phi(z) = (1 + \exp[-20(z - 0.5)])^{-1}. \tag{3}$$

The compatibility function is defined in terms of the scalar product between the evidence vectors of parts p_i and p_n,

$$C(p_i, p_n) = \vec{E}(p_i) \cdot \vec{E}(p_n). \tag{4}$$

For identical evidence vectors $\vec{E}(p_i)$ and $\vec{E}(p_n)$, $C(p_i, p_n) = 1$, and for incompatible evidence vectors, for example $\vec{E}(p_i) = [1, 0, 0]$ and $\vec{E}(p_n) = [0, 1, 0]$, $C(p_i, p_n) = 0$.

Compatibility of evidence vectors is a weak constraint for updating the evidence vectors of each part. Much stronger constraints can be derived from the comparison of test patterns to model patterns and the establishment of correspondence mappings between the two. This is further considered in the next section in the context of complex scene analysis.

4. COMPLEX SCENE ANALYSIS

As discussed before, identifying specific shapes in scenes composed of multiple objects or patterns cannot be directly accomplished using attribute-indexed pattern recognition systems such as neural networks or decision trees. Attempts to solve such problems using windowing or "perceptual grouping" methods are typically heuristic, unreliable and inefficient though, in some limited domains this can be of some use.[21,30] Here, we discuss a solution in the context of a rule-based system that makes only weak and general assumptions about the structure of scenes and objects. Our solution is based on the analysis of the relationships within (intra) and between (inter) instantiated rules generated by the CRG system. CRG has the advantage of actually "pre-compiling" the cliques or groupings of parts which, from the training data, are necessary and sufficient for discriminating between different models. The proposed solution method is based on the *sequential* evaluation of constraints described below.

Initial Rule Evaluation. The first stage involves direct activation of the CRG rules in a parallel, iterative deepening method. Starting from each scene part, all possible sequences of parts, termed *chains*, are generated and classified using the CRG rules. Expansion of each chain $S = \langle s_1, s_2, \ldots, s_n \rangle$ terminates if at least one of the following conditions occurs:

(1) the part sequence $s_1, s_2, \ldots, s_n$ cannot be expanded without creating a cycle,
(2) all CRG rules instantiated by S are completely resolved, or
(3) the binary features $\vec{b}(s_n, s_{n+1})$ do not satisfy the features bounds of any CRG rule.

If a chain S cannot be expanded, the evidence vectors of all rules instantiated by S are averaged to obtain the evidence vector $\vec{E}(S)$ of the chain S. Further, the initial evidence vector of a part p is obtained by averaging evidence vectors over the set $\mathcal{S}_p$ of all chains that start at p,

$$\vec{E}(p) = |\mathcal{S}_p|^{-1} \sum_{S \in \mathcal{S}_p} \vec{E}(S). \tag{5}$$

where $|\mathcal{S}_p|$ denotes the cardinality of the set $\mathcal{S}_p$. This initial estimate is improved in the next few steps by identifying "crossing" snakes and eliminating their contribution to the evidence vector $\vec{E}(p)$.

Chain Permutation Constraint. The chain permutation constraint is based on the assumption that rule instantiations are invariant to permutations, i.e. if two chains are permutations of each other, for example $S_1 = \langle A, B, C \rangle$ and $S_2 = \langle B, A, C \rangle$, their parts must index the same set of model parts, independent of chains and independent of instantiated rules.

Single Classification Constraint. The single classification constraint is based on the assumption that *at least one* chain among all chains starting at a scene part does not cross an object boundary and that at least one instantiated rule indexes the correct model parts. Given this, if there is any scene part that initiates a single chain S_i and this chain instantiates a single classification rule then the model parts indexed by S_i can be used to constrain all chains that touch S_i.

These two deterministic constraints are very powerful in terms of eliminating inconsistent (crossing) chains. Their usefulness breaks down, however, for cases where the assumptions are not met by training and test data sets.

Inter-chain Compatibility Analysis. The idea of the inter-chain compatibility analysis is as follows. The less compatible the evidence vector of a chain S_i is with the evidence vectors of all chains that S_i touches, the more likely it is that S_i crosses an object boundary. In this case $\vec{E}(S_i)$ is given a low weight in the computation of (5). Let S_i and S_j be touching chains, and let T_{ij} be the set of parts common to the two chains. A compatibility measure $C(S_i, S_j)$ between S_i and S_j can be defined based on the overlap of model parts instantiated by T_{ij} and the overall compatibility of a

chain S_i is then defined as the average compatibility of S_i with all chains that S_i touches:

$$w_{inter}(S_i) = |\mathcal{S}_T|^{-1} \sum_{S \in \mathcal{S}_T} C(S_i, S). \tag{6}$$

where $\mathcal{S}_T$ denotes the set of chains that touch S_i.

Intra-Chain Compatibility Analysis. The last rule for detecting boundary-crossing chains is based on the following idea. If a chain $S_i = \langle s_{i1}, s_{i2}, \ldots, s_{in} \rangle$ does not cross boundaries of objects then the evidence vectors $\vec{E}(s_{i1})$, $\vec{E}(s_{i2})$, $\ldots$, $\vec{E}(s_{in})$ are likely to be similar, and dissimilarity of the evidence vectors suggests that S_i may be a "crossing" chain. Similarity of any pair of evidence vectors can be measured by their dot product 4 and similarity of all intra-chain evidence vectors $w_{intra}(S)$ is obtained by averaging the evidence vectors of all pairs of parts in the chain S_i.

Evidence Combination. Combining inter- and intra-snake compatibilities, the evidence vectors of a part p can be computed using the following iterative relaxation scheme:

$$\vec{E}^{(t+1)}(p) = \Phi \left[Z^{-1} \sum_{S \in \mathcal{S}_p} w_{inter}(S) w_{intra}^{(t)}(S) \vec{E}(S) \right] \tag{7}$$

where Z is a normalizing factor and Φ the logistic function (3). Iterative computation of (7) is required since recomputation of $\vec{E}(p)$ affects the intra-chain compatibility. As indicated above, the four rules presented in this section are evaluated sequentially, and the final part classification is given by the iterative scheme (7).

For illustration purposes we consider the problem of identifying compound blocks in complex scenes.

4.1. Colored Blocks Example

Examples of isolated block configurations were presented during training in a supervised learning procedure (Fig. 6). Each block configuration consisted of three training examples, and the test arrangements of complex block combinations consisted of up to 20 blocks (Fig. 7).

Images of the training and test scenes were captured with a color camera. Preprocessing was fairly simple, consisting of a *segmentation* stage and a *feature extraction* stage. Segmentation was achieved using a form of K-means clustering on position (x, y) and color (r, g, b) attributes.[31] For the resulting clusters, small clusters were merged with larger neighbor clusters in order to eliminate spurious image regions. Given the rich image information, it is not surprising that the resulting image regions correspond fairly well to the individual blocks.

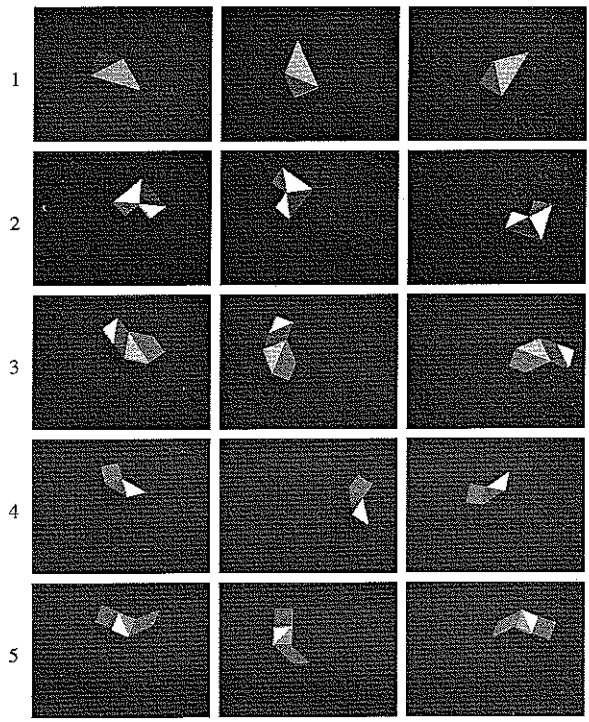


Fig. 6. Images of five classes of toy block configurations with three views each. The image parts are described by the unary features size, eccentricity and the three normalized color coordinates. Pairs of image parts are described by the binary features of midpoint distance, area-normalized midpoint distance, minimum distance and normalized shared boundary length.

In the feature extraction stage the following unary features were extracted for each image region: size (in pixels), compactness ($perimeter^2/area$), and the normalized color signals $R/(R+G+B)$, $G/(R+G+B)$, and $B/(R+G+B)$. For pairs of image regions the following binary features were computed: absolute distance of region centers, minimum distance between the regions, distance of region centers normalized by the sum of the region areas, and length of shared boundaries normalized by total boundary length.

For the training data, CRG analyzed 276 different chains of pattern parts and produced 32 rules: 9 *U*-rules, 4 *UB*-rules, 12 *UBU*-rules, 3 *UBUB*-rules, and 4 *UBUBU*-rules. From the distribution of rule types, it is evident that CRG used predominantly unary features for classification. Classification performance was tested on the complex configurations of block patterns, two of which are shown in Fig. 7 together with the classification results. Classification proceeded as described above, using the chain analysis and relaxation labeling solution. For both scenes, all parts (11 and 17 for the left and right scenes of Fig. 7, respectively) were classified correctly with the exception of a single part from the class-4 configuration.

For comparison purposes, we have analyzed the block example using classical decision trees.[32] In the first analysis, a *UBB*-triple analysis, each image part P of

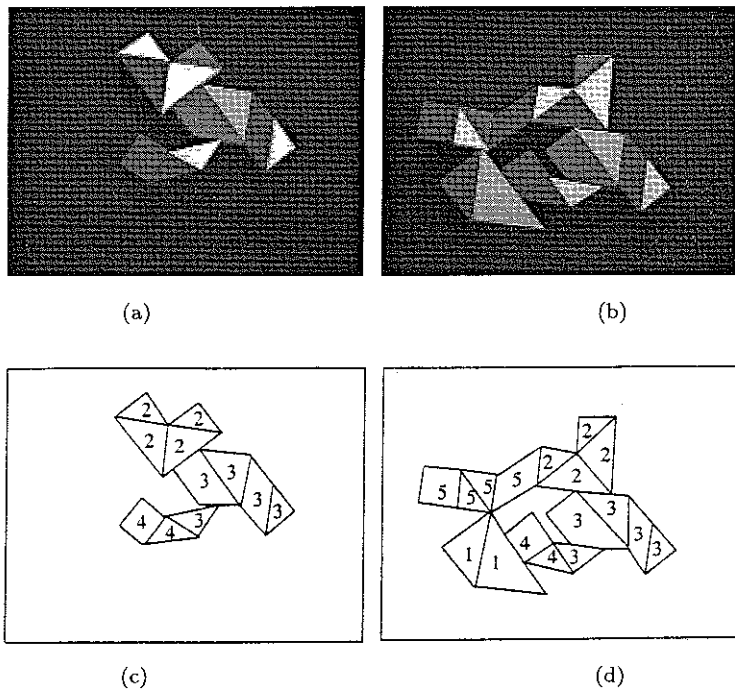


Fig. 7. Two block scenes and their classifications. (a) Block scene consisting of 11 blocks corresponding to examples of classes 2, 3, and 4. (b) Block scene consisting of 17 blocks corresponding to examples of all classes. (c) Classification result for block scene in (a) with region labels corresponding to classes. (d) Classification result for block scene in (b) with region labels corresponding to classes.

the training and test images was described by 13 features. These features consisted of the five unary features of P (see above), the four binary features (see above) of the relation between P and its closest neighbor, and another four binary features of the relation between P and its second-closest neighbor. For the class-1 cases which consisted of two parts only, the feature values for the second binary relation were set to "unknown". A decision tree was generated using C4.5 with default parameters,[32] and the resulting tree was used to classify all parts of the test scenes in Fig. 7. In each of the two scenes, three parts were misclassified. The good performance obtained with C4.5 is consistent with the observation that the use of higher-order relational information does not seem to be crucial for successful classification of this data set.

The first comparison using C4.5 employed features of all UBB-triples (unary features and binary features of relations with two other parts) for classification. A second analysis, using UBU-triples (with 14 features: the same five unary features of all pairs of parts, as well as the same four binary features of their relation) was performed, but the results cannot be interpreted as easily. For the scene in Fig. 7(a), 33 out of 110 UBU-triples or 30% were misclassified, and for the scene in Fig. 7(b), 103 out of 272 UBU-triples or 37.8% were misclassified. One reason for the error rate being so high is the fact that no analysis corresponding to the chain analysis

described above was performed with the C4.5 results. However, the error rates seem to be too high to be corrected using the relaxation scheme proposed there.

A general point is, however, more important. The CRG method generates rules of (minimal) variable length *optimized for a given training set*, whereas the decision tree (C4.5) *fixes* the dimensionality of the feature space and rule length. Indeed, C4.5 does not use part-indexing and so the relational structure has to be encoded implicitly in the attributes extracted from different parts and part relations. The choice of UBB-triples for the block example lead to a C4.5 performance that was essentially the same as that of CRG, but for the UBU-triples the C4.5 performance was much worse. This choice has to be done *a priori* whereas it is adjusted dynamically in the CRG method. Further, CRG is designed to exploit structural information of patterns and dependencies between feature states, whereas C4.5 analyzes a fixed set of features that are assumed to be independent. In this sense, the application of C4.5 to the blocks data was somewhat misleading in the sense that the necessary and relevant structural information was generated manually. This example of forcing C4.5 to function with RS and complex scene data emphasizes the very need for systems like CRG.

5. CONCLUSIONS

In this paper we have considered issues related to how knowledge can be used to develop strategies for the interpretation of image data. Although the examples have been visual, it is clear that such technologies can also be applied to other domains including the range of data types that exist in GIS, visualization and multi-media. Indeed, we claim that such technologies need to be explored for the development of content-based image queries of current interest in many commercial applications.

The main points of this paper are that ML has to be considered in the context of the data structures required to solve specific problems and the development of efficient and robust search procedures especially at the recognition stage. For example, if the data structure is a labeled and attributed graph then the aim of ML is to generalize on such relational structures and to find optimal subgraphs which are suited for evidencing one structure and not others. Equally, if the data structure is non-relational then standard attribute-indexed systems are sufficient.

Other ways in which knowledge can be introduced into visual learning resides in the notion of "model-based" processes. Even "unsupervised" learning models use this in so far as the spatial contiguity assumption, the cost functions and parameters used to model the clustering processes all assume data models. Equally, in the area of low-level vision, the modeling of windowing functions and the requirements for extraction of specific features are all examples of how specific domain knowledge and constraints are used in the generation of rules for interpreting image data.

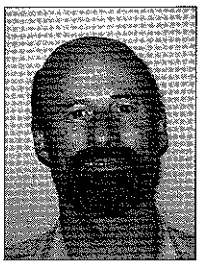
This perspective competes with "black-box" learning models where arbitrary attributes, decision functions are used to enable generalization from observed data. Ml offers essentially, a methodology for summarizing and generalizing from observed data and such processes have to be firmly embedded in the data structures or domain

knowledge types. Further work must also proceed on the integration of low and high-level learning modules where the performance of recognition or interpretation can be used to update even low-level feature extraction and attribute selection.

REFERENCES

1. P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, "AUTOCLASS: A Bayesian classification system", in *Proc. Fifth Int. Conf. on Machine Learning*, San Mateo, CA, 1988, pp. 54–64.
2. T. Poggio, "Networks for approximation and learning", *Proc. IEEE* **78** (1990) 1481–1497.
3. S. Amari and K. Maginu, "Statistical neurodynamics of associative memory", *Neural Networks* **1** (1988) 63–73; also found as a technical report at this site.
4. P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, "Using vector quantization for image processing", *Proc. IEEE* **81** (1993).
5. U. Fayyad and K. Irani, "On the handling of continuous-valued attributes in decision tree generation", *Machine Learning* **8** (1992) 87–102.
6. D. Fisher, "Knowledge acquisition via incremental conceptual clustering", *Machine Learning* **2**, 2 (1987) 139–172.
7. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, New York, 1973.
8. Y. Park, "A comparison of neural net classifiers and linear tree classifiers: Their similarities and differences", *Pattern Recognition* **27**, 11 (1994) 1493–1503.
9. S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture", in *Advances in Neural Information Processing Systems 2*, ed. D. S. Touretsky, Morgan Kaufmann, San Mateo, CA, 1990, pp. 38–51.
10. T. Caelli and A. Dreier, "Variations on the evidence-based object recognition theme", *Pattern Recognition* **27**, 2 (1994) 185-204.
11. J. Quinlan, "Induction of decision trees", *Machine Learning* **1** (1986) 81–106; reproduced in *Readings in Machine Learning*, Ref. 33, pp. 57–69.
12. A. K. Jain and R. Hoffman, "Evidence-based recognition of 3D objects", *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-10**, 6 (1988) 783–802.
13. W. F. Bischof and T. Caelli, "Learning structural descriptions of patterns: A new technique for conditional clustering and rule generation", *Pattern Recognition* **27**, 5 (1994) 689–697.
14. D. Gabor, "Theory of communication", *J. of the Institute of Electrical Engineers* **93**, 6 (1946) 429–459.
15. H. Kong and L. Guan, "A neural network adaptive filter for the removal of impulse noise in digital images", *Neutral Networks* **9** (1996) 373–378.
16. W. F. Bischof and T. Caelli, "Parsing scale-space and spatially stability analysis", *Computer Vision, Graphics and Image Processing* **42** (1988) 192–205.
17. B. McCane, T. Caelli, and O. de Vel, "Multi-scale adaptive segmentation using edge and region based attributes", submitted, 1996.
18. A. Jain and K. Karu, "Texture analysis: Representation and matching", in *Image Analysis and Processing, 8th Int. Conf. ICICAP '95*, eds., C. Braccini, L. DeFloriani, and G. Vernazza, San Remo, Italy, 1995, pp. 3–10.
19. T. Caelli, D. Squire, and T. Wild, "Model-based neural networks", *Neural Networks* **6** (1993) 613–625.
20. D. Squire and T. Caelli, "Shift, rotation and scale invariant signatures for two-dimensional contours in a neural network architecture", *Annals of Mathematics and Artificial Intelligence*, 1997 (in press).

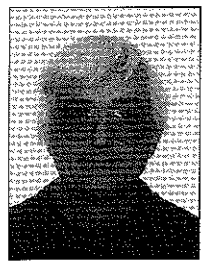
21. W. E. L. Grimson, *Object Recognition by Computer*, MIT Press, Cambridge, MA, 1990.
22. H. Bunke and B. Messmer, "Efficient attributed graph matching and its application to image analysis", in *Image Analysis and Processing, 8th Int. Conf. ICICA '95*, eds., L. DeFloriani and G. Vernazza, San Remo, Italy, 1995, pp. 45–55.
23. H. Bunke and B. Messmer, "Recent advances in graph matching", in *Spatial Computing*, eds., T. Caelli, H. Bunke, and P. Lam, World Scientific, Singapore, 1997.
24. P. J. Flynn and A. K. Jain, "BONSAI: 3D object recognition using constrained search", *IEEE Trans. on Pattern Analysis and Machine Intelligence* **13**, 10 (1991) 1066–1075.
25. T. Caelli and A. Pennington, "An improved rule generation method for evidence-based classification systems", *Pattern Recognition* **26**, 5 (1993) 733–740.
26. A. R. Pearce, T. Caelli, and W. F. Bischof, "Rulegraphs for graph matching in pattern recognition", *Pattern Recognition* **27**, 9 (1994) 1231–1247.
27. B. McCane, T. Caelli, and O. de Vel, "A fuzzy machine learning approach to recognizing 3D objects and 2D scenes", in *Proc. 2nd Asian Conf. on Computer Vision: ACCV '95*, vol. 3, 1995, pp. 106–111.
28. D. H. Wolpert, "Stacked generalization", *Neural Networks* **5** (1992) 241–259.
29. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, Orlando, Florida, 1982.
30. D. G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston, 1995.
31. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.
32. J. R. Quinlan, *C4.5 Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California, 1993.
33. J. W. Shavlik and T. G. Dietterich, eds., *Readings in Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1990.



Terry Caelli is Professor of computer science and Head of the School of Computing at Curtin University of Technology. Professor Caelli completed his Ph.D. at the University of Newcastle in the area of visual pattern recognition in 1975.

His current interests lie in computer vision, machine learning as they apply to building image interpretation and annotation systems for both static and dynamic spatial data and the inference of actions and function. Professor Caelli is actively involved in the International Association for Pattern Recognition and the IEEE and retains interests in understanding and modeling human vision.

His current application domains include human surveillance, environmental and industrial monitoring, and domain-specific multi-media query and annotation systems. Current research is funded via the Australian Research Council, Defence and local industries.



Walter F. Bischof received his Ph.D. degree in psychology in 1982 from the University of Bern, Switzerland. He is currently Associate Professor of psychology and Adjunct Professor of computer science at the University of Alberta and Adjunct Associate Professor of computer science at Curtin University.

His research interests are equally distributed between human and computer vision. In human vision, he works on early spatiotemporal analysis in the visual system with an emphasis on mechanisms of motion detection and motion integration. In computer vision his interests are focussed on object recognition and visual learning.